IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 19, Issue 4, Ver. III (Jul.-Aug. 2017), PP 50-55
www.iosrjournals.org

An Efficient General Decentralized Clustering Exploiting Hierarchy

*I.GeetaSowmya[1], K.NagaPrashanthi [2]
[1]*M.Tech.IV Sem.(CSE) Lakireddy Balireddy College of Engineering,Mylavaram*
[2]*Sr.Assistant Professor, CSE Department, Lakireddy Balireddy College of Engineering, Mylavaram*
Corresponding Author: I.GeetaSowmya

Abstract: *Clustering or unsupervised learning is important for analyzing large data sets. Large amounts of data are distributed among multiple sources. Examination of this data and identifying clusters is challenging due to processing storage and transmission costs. In this project we are implementing GD cluster, General Decentralized Clustering (GD) method which is capable of clustering dynamic and disturbed data sets. Nodes store and share a set of data items from other nodes. Data items represent internal data which may change over time and external data which may also store attribute vectors of data items from other nodes. Each data item is presented using an attribute vector. The union of internal and external data items is referred to as data item. Each node performs two tasks "DRIVE" and "COLLECT" which executes repeatedly and continuously in parallel. Nodes continuously cooperate through decentralized gossip-based communication to maintain summarized views of dataset. We modify GD cluster for execution of the Hierarchical Grid clustering methods on the summarized views and also offer enhancements to the basic algorithm. In this project experimental evaluation shows that GD cluster compared with the popular method LSP2P, clusters efficiently with scalable transmission cost.*

Indexterms: *Distributedsystems, clustering, hierarchical clustering, dynamic system*

Date of Submission: 04-07-2017 Date of acceptance: 26-07-2017

I. Introduction

Clustering can be a part of mainly unsupervised learning problem; so, as each problem of this kind, it deals with discovering a structure in a collection of unlabeled data. A loose definition of clustering could be "the process of organizing objects into groups whose members are similar in some way".A cluster is a gathering of objects which are "alike" among them and are "unlike" to the objects goingtootherclusters.We can show this with a simple graphical example in fig 1.

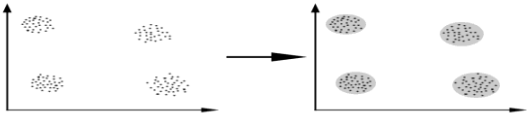

Fig 1: ASample Graphical view

In this case we easily identify the 4 clusters into which the data can be divided; the similarity criterion is distance: two or more objects belong to the same cluster if they are "near" according to a given distance. This is called distance-based clustering. One morekind of clustering is conceptual clustering: two or more objects belong to the same cluster if this one defines a concept common to all that objects. In other words, objects are grouped according to their fit to descriptive concepts, not according to simple similarity measures. So, the goal of clustering is to control the intrinsic grouping in a set of unlabeled data. But how to decide what creates a good clustering? It can be shown that there is no absolute "best" criterion which would be independent of the final aim of the clustering. Consequently, it is the user who must supply this standard, in such a way that the result of the clustering will suit their needs. We could be interested in finding representatives for homogeneous groups (data reduction), in finding "natural clusters" and describe their unknown properties ("natural" data types), in finding useful and suitable groupings ("useful" data classes) or in finding unusual data objects (outlier detection).A common approach in distributed clustering is to combine and merge local representations in a central node, or aggregate local models in a hierarchical structure.Some recent proposals, although being completely decentralized, include synchronization at the end of each round, and/or require nodes to maintain history of the clustering. In this paper, a general decentralized clustering algorithm (GDcluster) is proposed and instantiated with popular hierarchical clustering method. We first introduce a basic method in which nodes

gradually build a summarized view of the dataset by continuously exchanging information on data items and data representatives using gossip-based communication.Gossip is a simple communication which assumes no predefined structure in the networks. General Decentralized cluster dataset is circulated among large no of nodes in a distributed environment. GD cluster can cluster a data set which is dispersed among a large number of nodes in a distributed environment. It can handle the cluster named hierarchical grid cluster, while being fully decentralized, asynchronous, and also adaptable to churn. The general design principles employed in the proposed also allow customization for other classes of clustering, which are left out of the current paper. We also discuss enhancements to the algorithm particularly aimed at improving communication costs. The simulation results are presented using real-world distributed systems change continuously, because of nodes joining and leaving the system, or because their set of internal data is modified. GD cluster is able to achieve a high-quality global clustering solution, which approximates centralized clustering. We also explain the effects of various parameters on the accuracy and overhead of the algorithm. We compare our proposal with central clustering and with the large scale peer-to-peer(LSp2p) network algorithm. If data sources are dispersed over a large scale peer-2-peer network, collecting the data at a central location before clustering is not an attractive and practical option.

The main contributions of this paper are as follows:

- Proposing a new fully distributed clustering algorithm, which can be instantiatedwith hierarchy.
- Dealing with dynamic data and evolving the clustering model.
- Empowering nodes to construct a summarized view of the data, to be able to execute a customized clustering algorithm independently.

II. Decentralized Clustering

Assumingthat the entire data set can be summarized in each node p, by means of representatives. Each node p is responsible for deriving accurate representatives for part of the dataset located near internal data. For other parts, it solely collects representatives. Accordingly, it gradually builds a global view of data items. Each node continuously performs two tasks in parallel: i) representative derivation, namely DERIVE fig 2(a). ii) representative collection, namely COLLECT fig 2(b). The two tasks can execute repeatedly and continuously in parallel. The details of two tasks are explained below.

a) DERIVE

To derive representatives for part of the data set located near internal data, node p should have an accurate and up-to-date view of the data located around each data belongs to internal data. In each round of DERIVE task, each node p selects another node q for three-way information exchange.It should first send internal data to node q. if size of internal data is large, it can summarize the internal data by an arbitrary method such as grouping the data using clustering, and sending one data from each group. Node p then receives from q, data items located in radius of each data belonging to internal data, based on a distance function. Radius is a user-defined threshold, which can be adjusted as p continues to discover data in the same manner, it will also send to q the data in data node p that lie within the radius of data in internal data of node q. The operation updateLocalData( ) is used to add the received data to internal data of node p.Knowing some data located within radius of some internal data item d, node p can summarize all this data into one representative. This is performed periodically every gossip rounds using the algorithm. The mergeWeights function, updates the representative weight, and is later described.

b)*COLLECT*

To achieve the COLLECT task, each node p selects a random node every T time units, to exchange their set of representatives with each other. Both nodes store the full set of representatives. The summarized function used in the algorithm, simply returns all the representatives. The summarize function used in the algorithm, simply returns all the representatives given to it as input. A special implementation of this function is described. Which reduces the number of representatives at each node is initialized with all of its data items. The two algorithmsof tasks DERIVE and COLLECT, start with a preprocessing operations and have no special function, thus we defer of the communication performed in DERIVE and COLLECT.

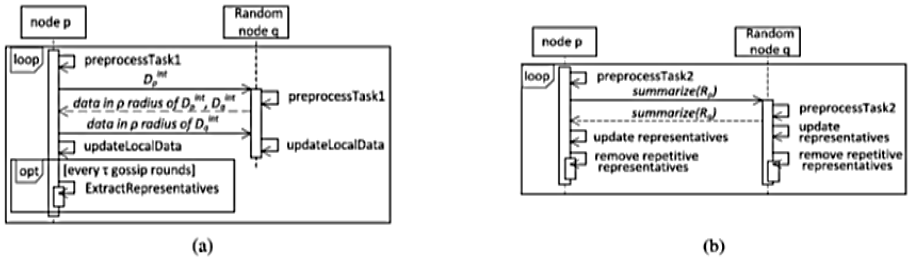

FIG 2 (a): Decentralized clustering **DERIVE**FIG 2 (b): Decentralized clustering **COLLECT**

The operation selectNode ( ) employs a peer-sampling service to return a node selected uniformly at random from all live nodes in the system.

2.1 Gossip communication

In tasks DERIVE and COLLECT we use gossip communication as a propagation media. This is in particular different from aggregation protocols [8] which employ gossiping to reach consensus on aggregations. Using information in [8] and turning a blind eye to the Details, the general approach of GD cluster can be simplified as follows. At all times a node p maintains an ordered set(not a sum) $s_{t,p}$ initialized to s0,p = internal data, and an ordered set of corresponding weights $w_{t,p}$. At each time step $_{t,p}$ chooses a target node ft(p) uniformly at random and sends both collections to that node and itself. It calculates union of the pairs from received pairs from other nodes with its own s and n sets. In step t of the algorithm, st,p is view p has on the entire dataset, while wt,p contains the corresponding weight of each view element. As the set s quickly becomes large, the notion of representatives are introduced. Node *p* can summarize the elements of $s_{t,p}$. The corresponding weights should also be removed and replaced by the aggregate weight. This summarized view is labeled R*p* in this paper. Accroding to [8] and [10], a message that orginates with p at time $t_0$and is forwarded by all nodes that have received it, will reach all nodes in time at most 4 logN + log 2/δ with probability at least 1-δ/2. Therefore, after the same time order, the summarized view of *p*, will have elements from all other nodes, either in their raw form or embedded in a representative.

Compare LSP2P

The LSP2P algorithm [4] executes the k-means in an iterative manner, with each node synchronizing with its neighbors during each iteration. In a static setting, the algorithm is initiated at a single node p, which picks a set of random initial centroids along with a termination thresholdΓ > 0 (which we explain shortly). P sends these to all its immediate neighbors, and begins iteration 1. When a node receives the initial centroids and threshold for the first time, it forwards them to its remaining neighbors and initiates iteration 1. In each iteration, every node p executes one round of k-means on its local data based on the centroids computed in the previous iteration. It then prompts its immediate neighbors for their corresponding cluster centroids, and updates local centroids based on the received information. Once the computed centroids of two consecutive iterations, deviate less than Γ from each other, p enters the terminated state. In dynamic setting, the change of data may reactivate the nodes.

III. Hierarchical Grid Clustering

- A hierarchy can relate entities either directly or indirectly, and either vertically or diagonally.
- A **hierarchy** is scheme or organization in which people or collections are graded one above the other rendering to status or authority.
- The *grid-based clustering* method uses a multi resolution grid data structure.
- It quantizes the entity space into a limited number of cells that form a grid structure on which all of the actions for clustering can be performed.
- Main advantage of the approach is its fast processing time, which is typically independent of the number of data objects, thus, dependent on only the number of cells.

3.1 ALGORITHM

- Begin with the disjoint clustering having level L(0) = 0 and sequence number m = 0.
- Find the least dissimilar pair of clusters in the current clustering,$say\ pair\ (r),(s)$, according to d[(r),(s)]=mind[(i),(j)] where the minimum is over all pairs of clusters in the current clustering.
- Increment the sequence number: m = m +1. Merge clusters (r) and (s) into a single cluster to form the following clustering m. Set thelevelofthisclusteringtoL(m) = d[(r),(s)].
- Renew the closeness of matrix, D, by deleting the rows and columns consistent to clusters (r) and (s) and adding a row and column corresponding to the newly formed cluster. The proximity between the new cluster, denoted (r,s) and old cluster (k)isdefinedinthisway:d[(k), (r,s)] = min d[(k),(r)], d[(k),(s)]
- If all objects are in one cluster, stop. Else, go to step 2.

3.2 EXAMPLE

- Consider a disjoint clustering with the following values

	p1	p2	p3	p4	p5	p6
p1	0.00	0.24	0.22	0.37	0.34	0.23
p2	0.24	0.00	0.15	0.20	0.14	0.25
p3	0.22	0.15	0.00	0.15	0.28	0.11
p4	0.37	0.20	0.15	0.00	0.29	0.22
p5	0.34	0.14	0.28	0.29	0.00	0.39
p6	0.23	0.25	**0.11**	0.22	0.39	0.00

Table 1: values of disjoint clustering.

The minimum distance clusters among all the given is 3 and 6 dist=0.11 and we group 3 and 6 into single cluster and find the distance of all other clusters to newly formed cluster.

- The formula for finding distance is as follows:

d[(k), (r,s)] = min d[(k),(r)], d[(k),(s)]
Here k=1 and r=3 and s=6

d[(1), (3,6)] = min d[(1),(3)], d[(1),(6)]
min d[0.22,0.23]=0.22
k=2
min d[0.15,0.25]=0.15
k=4
min d[0.15,0.22]=0.15
k=5
min d[0.28,0.39]=0.28

Newly formed matrix is as follows

	p1	p2	P3,6	p4	p5
p1	0.00	0.24	0.22	0.37	0.34
p2	0.24	0.00	0.15	0.20	0.14
P3,6	0.22	0.15	0.00	0.15	0.28
p4	0.37	0.20	0.15	0.00	0.22
p6	0.34	**0.14**	0.28	0.29	0.00

Table 2: Newly formed matrix

In the same we group all the remaining clusters into single cluster by finding the least distance between them. Next least distance 2 and 5 dist = 0.14 the matrix is as follows:

	p1	P2,5	P3,6	p4
p1	0.00	0.24	0.22	0.37
P2,5	0.24	0.00	0.15	0.15
P3,6	0.22	0.15	0.00	0.15
p4	0.37	**0.15**	0.15	0.00

Table 3: Least distance for 2and 5 with dist=0.14

- Next least distance is 3,6 and 4 dist=0.15 the matrix is as follows:

	p1	P2,5	P3,4,6
p1	0.00	0.24	0.22
P2,5	0.24	0.00	0.15
P3,6,4	0.22	**0.15**	0.00

Table 4: least distance for 3,6 and 4 with dist=0.15

Next least distance is 2,5 and 3,4,6 dist=0.15 the matrix is as follows:

	p1	P2,3,4,5,6
p1	0.00	0.22
P2,3,4,5,6	0.22	0.00

Table 5: The minimum distance of cluster is 0.22.

IV. .System Architecture

In this paper we consider N number of nodes. We select a sender and destination node so that we send and receive packets. We create a dataset based on the packets received by the destination node from the sender and we apply clustering on the dataset to remove unnecessary attributes to reduce complexity and finally compare results. This is shown in fig 4.

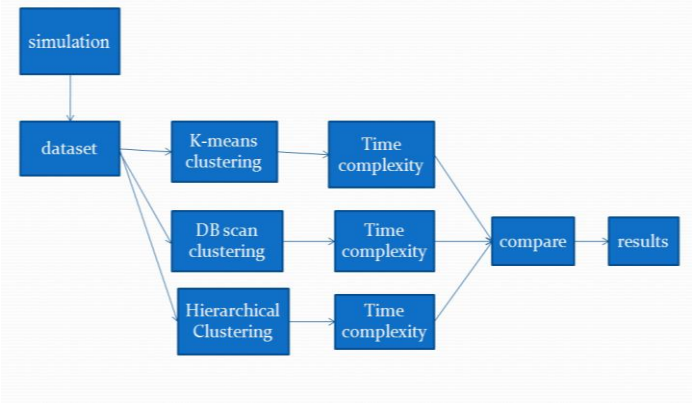

FIG 4: System Architeture

4.1 DYNAMIC DATA SET

Real-world distributed systems change continously,because of nodes joining and leaving the system, or because their set of internal data is modified. To avoid the hardest of data, each data item will have an associated age. When the data item reaches the particular associated age the data item is discarded automatically so that we can avoid duplicate values where the derive and collect maintains a summarized view every node present. Max age is calculated through thershold value and time is measured interms of gossiped rounds.

V. .Weight Calculation

When representatives are merged, for example, in the function removeRepetetives, a special method should be devised for weight calculation. The algorithm of hierarchical grid clustering we consider level from 0 and sequence number m=0 now we take a dissimilar pair of clusters say r, s we calculate the minimum distance points among all the points in the given dissimilar cluster using following formula .The formula for finding minimum distance is as follows:

d[(r),(s)] = min d[(i),(j)]

Now, we increment the sequence number and set the level using following formula

The formula for setting the level is as follows:

L(m) = d[(r),(s)]

Now, we update the proximity measure by deleting the rows and columns.The old cluster is denoted using formula.

The formula used to represent the updated matrix is as follows:

d[(k), (r,s)] = min d[(k),(r)], d[(k),(s)]

until only one cluster remains.

VI. Performance Evaluation

We evaluate the GDCluster algorithm in static and dynamic settings. We will also compare GDCluster with a central approach and with LSP2P, a recently proposed algorithm being able to execute in similar distributed settings.

6.1 EVOLUTION MODEL

We consider a system of N nodes, each node initially holding a number of data items, and carrying out DERIVE and COLLECT tasks iteratively. For simplicity and better understanding of the algorithm, we consider only data items. By using the peer sampling service, the network structure is not a concern in the evolutions [9]. Each cluster in the synthetic data sets consists of a skewed set of data composed from two Gaussian distributions with different values of mean and standard deviation. The real datasets used for the hierarchical grid clustering is well-known shuttle. The shuttle dataset contains only numerical values thathas 9 attributes. From data set, a random sample 10,240 instances are used in the experiments. To assign the data set D to nodes, two data-assignment strategies are employed, which aid at revealing special behaviors of the algorithm:

Random data assignment (RA): each node is assigned data randomly chosen from D.

Cluster-aware data assignment (CA): each node is assigned data from a limited number of clusters.

VII. Related Work

Distributed data mining is a dynamically growing area. Adiscussion and comparison of several distributed centroidbased partitional clustering algorithms is provided in reference propose parallel K-means clustering, by first distributing data to multiple processors. In each synchronized algorithm round, every processor broadcasts its currently obtained centroids, and updates the centroids based on the information received from all other processors. Different from many existing distributed clustering algorithms, our algorithm does not require a central site to coordinate execution rounds, and/or merge local models. Also, it avoids global message flooding. RACHET is a

hierarchical clustering algorithm in which, each site executes the clustering algorithm locally, and transmits a set of statistics to a central site. A distributed partition-based clustering algorithm for clustering documents in a peer-to-peer network is proposed by Eisenhardt et al.The algorithm requires rounds of information collection from all peers in the network. A K-means monitoring algorithm is proposed in this algorithm executes K-means by iteratively combining data samples at a central cite, and monitoring the deviation of centroids in a distributed manner.

VIII. Conclusion

In this paper we first identified the necessity of an effective and efficient distributed clustering algorithm. Dynamic nature of data demands a continuously running algorithm which can update the clustering model efficiently, and at a reasonable place. We introduced GDCluster, a general fully decentralized clustering algorithm, and instantiated it for hierarchical grid clustering methods. The proposed algorithm enabeld nodes to gradually build a summarized view on the global data set, and execute weighted clustering algorithm to build the clustering models. Adaptability to dynamics of the data set was made possible by introducing an age factor which assisted in detecting data set changes updating the clustering model.Our experimental evolution proves that hierarchical grid clustering can better satisfy the specific requirementsofdistributed systems.Embeddedflexibility regarding the level of granularity. And the algorithm is well suited for problems involving point linkages.

References

[1] K. M. Hammouda and M. S. Kamel, "Models of distributed data clustering in peer-to-peer environments," Knowl. Inf. Syst., vol. 38, no. 2, pp. 303–329, 2014.

[2] E. Januzaj, H.-P. Kriegel, and M. Pfeifle, "Scalable density-based distributed clustering," in Proc. 8th Eur. Conf. Principles Pract. Knowl. Discovery Databases, 2004, pp. 231–244.

[3] S. Lodi, G. Moro, and C. Sartori, "Distributed data clustering in multi-dimensional peer-to-peer networks," in Proc. 21st Austral-asian Conf. Database Technol., 2010, vol. 104, pp. 171–178.

[4] S. Datta, C. R. Giannella, and H. Kargupta, "Approximate distrib-uted k-means clustering over a peer-to-peer network," IEEE Trans. Knowl. Data Eng., vol. 21, no. 10, pp. 1372–1388, Oct. 2009.

[5] A. Elgohary and M. A. Ismail, "Efficient data clustering over peer-to-peer networks," in Proc. 11th Int. Conf. Intell. Syst. Des. Appl., 2011, pp. 208–212.

[6a] G. Di Fatta, F. Blasa, S. Cafiero, and G. Fortino, "Epidemic k-means clustering," in Proc. Int. Conf. Data Min. Workshops, 2011, 151–158.

[7] J. Fellus, D. Picard, and P.-H. Gosselin, "Decentralized k-means using randomized gossip protocols for clustering large datasets," in Proc. Data Min. Workshops, 2013, pp. 599–606.

[8] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in Proc. 44th Symp. Found. Comput. Sci., 2003, pp. 482–491.

[9] M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, and M. van Steen, "Gossip-based peer sampling," ACM Trans. Comput. Syst., vol. 25, no. 3, article 8, Aug. 2007.

[10] A. M. Frieze and G. R. Grimmett, "The shortest-path problem for graphs with random arc-lengths," Discr. Appl. Math., vol. 10, no. 1,57–77, 1985.

[11] D. Mosk-Aoyama and D. Shah, "Computing separable functions via gossip," in Proc. 25th ACM Symp. Principles Distrib. Comput., 2006, pp. 113–122.

[12] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in Proc. 5th Berkeley Symp. Math. Stat-ist. Probability, 1967, vol. 1, no. 14, pp. 281–297.

[13] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in Proc. 2nd Int. Conf. Knowl. Discovery Data Min., 1996,226–231.

[14] H. Mashayekhi, J. Habibi, S. Voulgaris, and M. van Steen, "Goscan: Decentralized scalable data clustering," Computing, vol. 95, no. 9, pp. 759–784, 2013.

IOSR Journal of Computer Engineering (IOSR-JCE) is UGC approved Journal with Sl. No. 5019, Journal no. 49102.

I.GeetaSowmya. "An Efficient General Decentralized Clustering Exploiting Hierarchy." IOSR Journal of Computer Engineering (IOSR-JCE) 19.4 (2017): 50-55.